CS 38000 Artificial Intelligence Name_____
Assignment 01

This assignment As 01 is at 1:30 pm. (in class), Tuesday, February 11, 2025. Please submit your solutions for the assignment at the beginning of the class. Do not submit your solution on Brightspace. Late submissions will not be accepted. Please write your name (first name followed by last name) on the first page of your assignment. Please number your problem-answer clearly, such as Problem 1(a), 1(b), 1(c), 2(a), 2(b), 2(c), 2(d), 2(e), etc. Arrange your answers in the same order as the problems are given. I strongly prefer you type your solutions instead of handwriting them.

The total number of points for this Assignment_01 is 100 points.
You need to provide details and explanations of your work, demonstrating how you obtained the solution for each problem.

General Description
Term Project: (This term project will be divided into several modules (i.e., assignments). We will employ what we have learned from the lecture notes to design and implement an intelligent agent that traverses the given park from its entrance through an exit gate. Our environment is referred to as the given park. For a given park, we create a maze associated with the park. Then, we label the maze in several locations. Based on the obtained maze with labels, we develop a graph in which the nodes are one-to-one correspondence between the nodes and labeled locations in the maze. For a path between two locations in the given park, there corresponds to an avenue between two locations in the maze, and there is an edge between two nodes. All the labels in the maze are *necessary* and *sufficient* so that the maze (or labyrinth) and the graph are one-to-one correspondence. Therefore, the constructed graph represents the given park concisely. Should the maze serve as a vehicle for translating the park into a graph? We are not interested in using any existing implemented maze problem solver.

Project description:
Consider the problem of developing (designing and implementing) an intelligent agent (also called an agent or the maze problem-solver). This agent can perceive (S) its environment (E) in the park at time intervals, forming a possible sequence of percepts. Based on the percepts, it can map to a possible action (A) the agent can perform (P) based on the predefined decision rules.

Given ANY maze configuration, the maze has an entrance gate (equivalently, called the initial state) for an agent entering the maze and reaching any of the two exit gates, g1 and g2 (or final states). The maze problem-solver can find its way from an entrance **[ ]** to an exit **] [.** The goal is for the maze problem-solver to enter the given maze and find a way to exit the maze with a minimal time span (in terms of the number of squares traversed).

At the end of this assignment, we are given a maze configuration. We will continue to use this maze configuration for the remaining assignments (i.e., project). This project is to program an

agent called the maze problem-solver, which will run around any given maze to find its way out from an exit gate. We will address the maze configuration and its representation later, which could serve as input to the program.

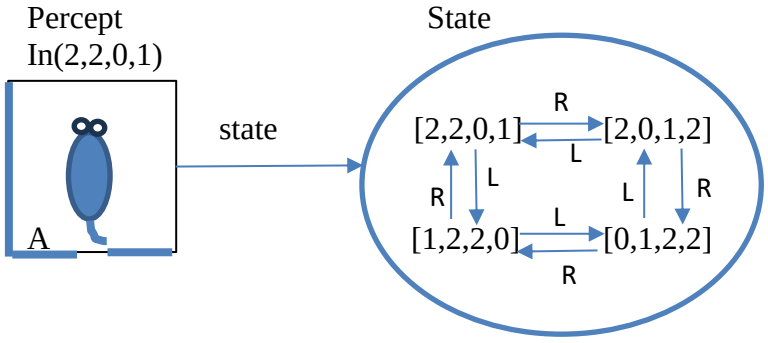
Assignment 01 consists of three questions, Q1 through Q5.

Q1:

An *agent program* can be defined as an agent function, $f: \mathcal{P}^* \rightarrow \mathcal{A}$, which maps every *possible* percept sequence to *a possible* action the agent can perform.

Let a grid in the maze be a percept of perceiving the environment at a particular time interval. Assume that the state representation for each of the grids is as follows:

Let the wall line (obstacle) have the value 1, the grid line (open) is 2, and the exit line (with a gate) is 0. For each grid, we will use these values to describe the four sides of any grid: the **t**op (the head of the agent), the **r**ight, the **b**ottom (its tail or back), and the **l**eft. At any grid, a corresponding percept can be represented at In(t, r, b, l) to form a state [t, r, b, l]. This also coincides with the fact that the head of the agent is pointing to the top of a grid, and the agent's tail is pointing at the bottom square. The specification of a percept and its state is a one-to-one correspondence. For example, if the agent enters the entrance gate, the agent is in the first grid, say A



The agent enters the grid, which receives the percept In(2, 2, 0, 1). This describes that the agent at the entrance grid has an open front and right side, a gate at its back (or tail), and a wall (or obstacle) at the left side of the agent. At this grid, the agent turns right to generate the percept In(2, 0, 1, 2), turns another right again to yield the percept In(0, 1, 2, 2), turns another right again to yield the percept (1, 2, 2, 0), returns to the initial position with the percept In(2, 2, 0, 1). The left turn of the agent at the grid will yield the opposite sequence of percepts. Thus, corresponding to this grid, a State consists of a sequence of these four precepts with right and left turns. The State describes the agent's location and position. This State is shown in the figure above. For the sake of simplicity, the grid can also be represented as In(A), where the grid is labeled as A.

The specification of the percept In(2, 2, 0, 1) can be simplified as [2, 2, 0, 1]. Therefore, using the specification [t, r, b, l] to represent the percept In(2, 2, 0, 1) suffices. For this reason, we will use the State's specification to represent percepts. If we use In(A) as an initial state, then the state can be described as [2, 2, 0, 1]. However, if we insist the agent should be outside the maze facing the entrance, then the initial state [0, -, -, -] or [0, 2, 2, 2] where the precept is In(0, -, -, -) or In(0, 2, 2, 2), respectively. All these descriptions are consistent with the Simple-Reflex-Agent(percept). It states

State ← INTERPRET_INPUT(percept)

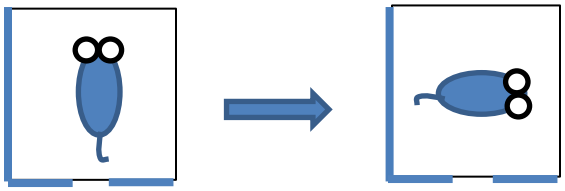
where percept is the grid's physical configuration generated by the agent sensor. It means that the image is converted into a program variable, State, using the function INTERPRET_INPUT.

The agent has three actions: rightTurn, leftTurn, and forward.

The rightTurn (R) within a grid is defined as follows:

rightTurn(In(2, 2, 0, 1), In(2, 0, 1, 2)). {brings the first component to the last; in fact. "In" can be omitted}

This means the agent is first in a grid heading in a direction. The right-turn action enables the agent to remain in the same grid, but the agent turns right, heading in a new direction that is perpendicular to the initial direction. Pictorially,



[2, 2, 0, 1] [2, 0, 1, 2].

Using the State's specifications, it is **rightTurn**([2, 2, 0, 1], [2, 0, 1, 2]), or

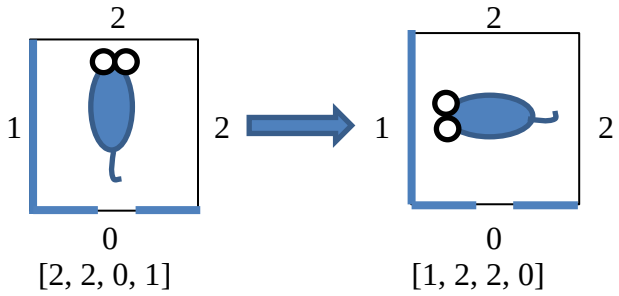
R

[2, 2, 0, 1] → [2, 0, 1, 2].

The leftTurn(L) within a grid is defined as follows:

leftTurn(In(2, 2, 0, 1), In(1, 2, 2, 0)). {brings the last component to be the first}

This means the leftTurn action enables the agent to remain in the same grid but in a new direction of the agent that is perpendicular to the previous (initial) direction and faces to the left side of the initial direction. Pictorially



[2, 2, 0, 1] [1, 2, 2, 0]

Using the State's specifications, it is **leftTurn**([2, 2, 0, 1], [1, 2, 2, 0]), or

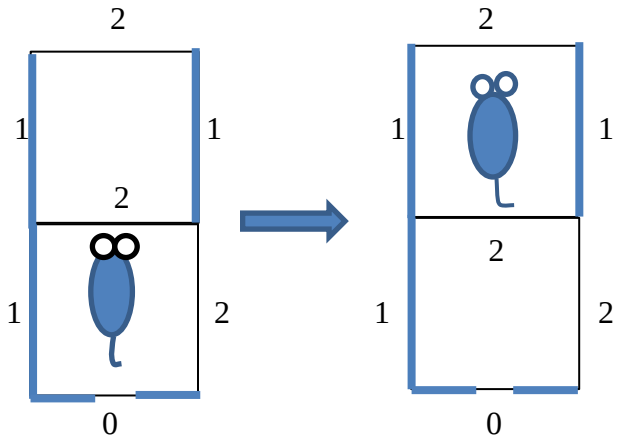
L

[2, 2, 0, 1] → [1, 2, 2, 0]

The forward (F) for two adjacent grids is defined as follows:

For the following pictorial example, we can describe the agent forwards from one grid to another with different percepts. This first parameter describes that the agent is in the first grip and then goes into the other grip with two walls and two grid lines. We use forward action to describe the following:

forward(In(2, 2, 0, 1), In(2, 1, 2, 1))



Using the state's specifications, it is **forward**([2, 2, 0, 1], [2, 1, 2, 1]), or

F

[2, 2, 0, 1] → [2, 1, 2, 1]

Using the above description, give/describe your proposed percepts and actions. (Warning: the later assignments will be based on your previous work! Therefore, a good and detailed work would help you with the rest of your project (i.e., assignments).

(a) How many different configurations of grids are in the given maze?
(b) How many different actions are based on the given maze?
(c) Specify its State description corresponding to each of the distinct grids.

Q2:

For designing a maze problem solver (or an intelligent agent) that could act rationally, the rationality at any given time depends on the PEAS. Define the PEAS for this maze problem solver:

Performance measure?
Environment? Complete the following form.

Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents

Actuators:
Sensors

Q3:

There are several agent architectures, which are as follows:

- Table-driven agents
- reflex
- reflex with state
- goal-based
- utility-based
- learning

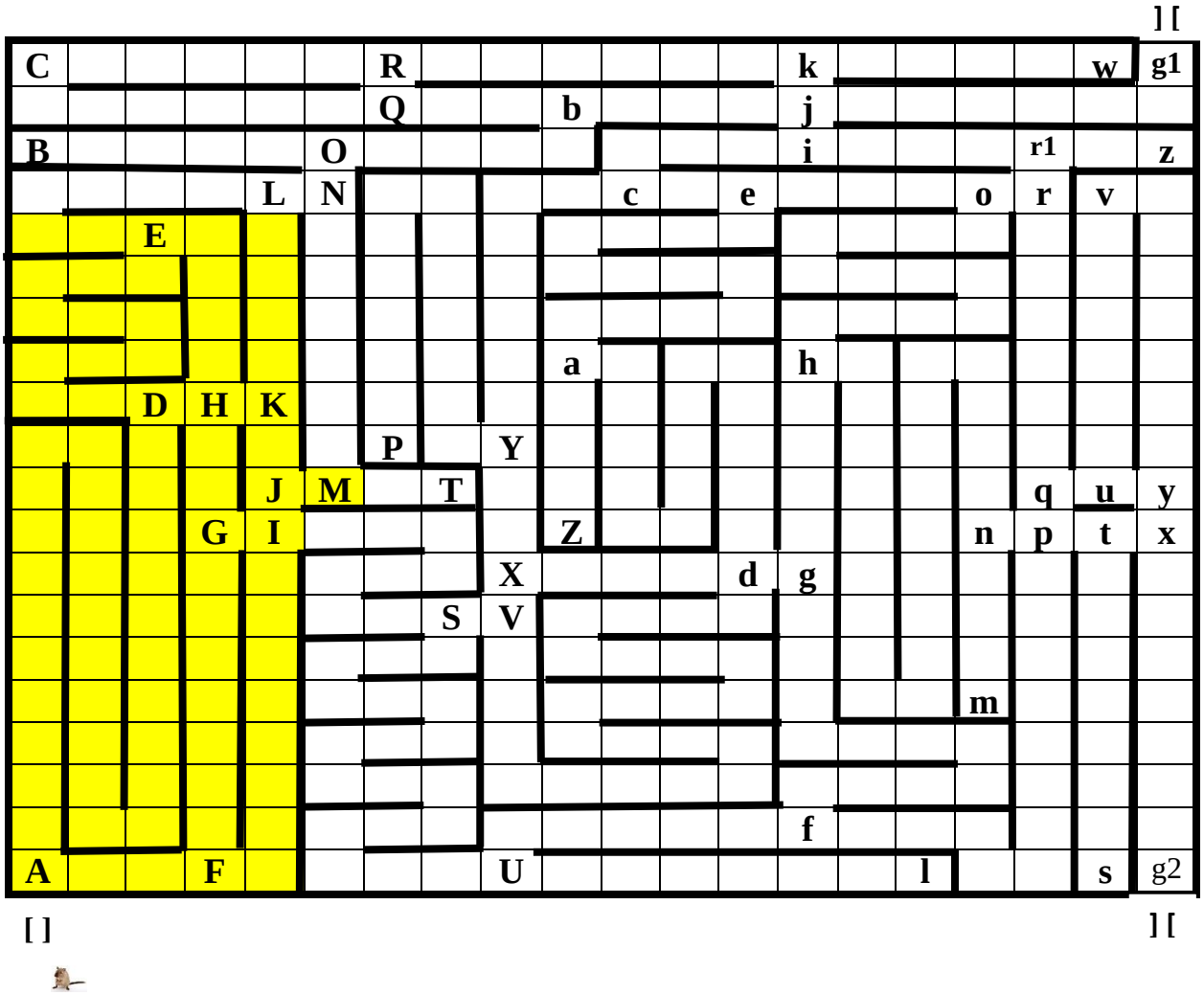
Which agent architecture would you like to use? Give your reasons.

Q4:

(a) Convert the given maze into a graph. (Hint: For simplicity's sake, use those specified grids (labeled with an alphabet) to name the nodes; the distance between two pairs of labeled nodes is the number of grids between the corresponding pairs of labeled grids as their distance. For example, the nodes C and R are reachable from each other by forwarding six (6) grids from C towards R or R towards C.)
(b) Does the obtained undirected, weighted, and labeled graph satisfy the necessary and sufficient requirement?
(c) Show that grid C requires labeling.
(d) Show that grid r1 and grid r require labeling.
(e) Show that grid H requires labeling.
(f) Show that grid A requires labeling.

Q5:
Considering a portion of the graph for the yellow-colored part of the maze, design a state search space in terms of States, state-actions, forward actions, and directed edges.

The following graph is a maze example: Assume A is the entrance [ ]. Both g1 and g2 are exits] [.



Notes:

Future Research for your term project: (I will discuss the following in class and will take time to develop its description):

Develop a necessary and sufficient theory to claim annotations, A, B, …, Z, a, b, …, z. Obtain a connected undirected weighted graph from a given maze. Will this obtained graph uniquely correspond to the given maze? (e.g., consider {v, q, u, y} against {s, p, t, x} for the unique paths).

Design and implement the agent (the maze problem-solver, an agent, a program, or a simulator) such that for any maze, the agent can find an exit efficiently once it enters the maze. Input to the agent is any maze. (The agent can take different mazes as input.) The agent consisting algorithms that traverse a given maze to find an exit.

/CS 380 As01 Q 02032025F